TriDet: Temporal Action Detection with Relative Boundary Modeling

Dingfeng Shi*
VRLab, Beihang University, China
shidingfeng@buaa.edu.cn

Yujie Zhong
Meituan Inc.
jaszhong@hotmail.com

Qiong Cao†
JD Explore Academy
mathqiong2012@gmail.com

Lin Ma
Meituan Inc.
forest.linma@gmail.com

Jia Li†
VRLab, Beihang University, China
jiali@buaa.edu.cn

Dacheng Tao
JD Explore Academy
dacheng.tao@gmail.com

Abstract

In this paper, we present a one-stage framework TriDet for temporal action detection. Existing methods often suffer from imprecise boundary predictions due to the ambiguous action boundaries in videos. To alleviate this problem, we propose a novel Trident-head to model the action boundary via an estimated relative probability distribution around the boundary. In the feature pyramid of TriDet, we propose an efficient Scalable-Granularity Perception (SGP) layer to mitigate the rank loss problem of self-attention that takes place in the video features and aggregate information across different temporal granularities. Benefiting from the Trident-head and the SGP-based feature pyramid, TriDet achieves state-of-the-art performance on three challenging benchmarks: THUMOS14, HACS and EPIC-KITCHEN100, with lower computational costs, compared to previous methods. For example, TriDet hits an average mAP of 69.3%, outperforming the previous best by 2.5%, but with only 74.6% of its latency. The code is released to https://github.com/dingfengshi/TriDet.

1. Introduction

Temporal action detection (TAD) aims to detect all start and end instants and corresponding action categories from an untrimmed video, which has received widespread attention. TAD has been significantly improved with the help of the deep learning. However, TAD remains to be a very challenging task due to some unresolved problems.

A critical problem in TAD is that action boundaries are usually not obvious. Unlike the situation in object detection where there are usually clear boundaries between the objects and the background, the action boundaries in videos can be fuzzy. A concrete manifestation of this is that the instants (i.e. temporal locations in the video feature sequence) around the boundary have relatively higher predicted response value from the classifier.

Some previous works attempt to locate the boundaries based on the global feature of a predicted temporal segment [21,22,29,46,51], which may ignore detailed information at each instant. As another line of work, they directly regress the boundaries based on a single instant, potentially with some other features. Ours: the action boundaries are modeled via an estimated relative probability distribution of the boundary.
Temporal action detection. Temporal action detection (TAD) involves localizing and classifying all actions from an untrimmed video. The existing methods can be roughly divided into two categories, namely, two-stage methods and one-stage methods. The two-stage methods [33, 36, 43, 46, 53] split the detection process into two stages: proposal generation and proposal classification. Most of the previous works [8, 13, 19, 21, 22, 26] put emphasis on the proposal generation phrase. Concretely, some works [8, 21, 22] predict the probability of the action boundary and densely match the start and end instants according to the prediction score. Anchor-based methods [13, 19] classify actions from specific anchor windows. However, two-stage methods suffer from a high complexity problem and can not be trained in an end-to-end manner. The one-stage methods do the localization and classification with a single network. Some previous works [20, 44, 45] build this hierarchical architecture with the convolutional network (CNN). However, there is still a performance gap between the CNN-based and the latest TAD methods.

Object detection. Object detection is a twin task of TAD. General Focal Loss [18] transforms bounding box regression from learning Dirac delta distribution to a general distribution function. Some methods [10, 15, 28] use Depth-wise Convolution to model network structure and some branched designs [16, 37] show high generalization ability. They are enlightening for the architecture design of TAD.

Transformer-based methods. Inspired by the great success of the Transformer in the field of machine translation and object detection, some recent works [9, 25, 27, 35, 38, 47] adopt the attention mechanism in TAD task, which help improve the detection performance. For example, some works [27, 35, 38] detect the action with the DETR-like Transformer-based decoder [6], which models action instances as a set of learnable. Other works [9, 47] extract a video representation with a Transformer-based encoder. However, most of these methods are based on the local behavior. Namely, they conduct attention operation only in a local window, which introduces an inductive bias similar to CNN but with a larger computational complexity and additional limitations (e.g. The length of the sequence needs to be pre-padded to an integer multiple of the window size.).

3. Method

Problem definition. We first give a formal definition for TAD task. Specifically, given a set of untrimmed videos \( \mathcal{D} = \{ V_i \}_{i=1}^n \), we have a set of RGB (and optical flow)
temporal visual features $X_i = \{x_i^t\}_{t=1}^T$ from each video $V_i$, where $T$ corresponds to the number of instants, and $K_i$ segment labels $Y_i = \{s_k, c_k, e_k\}_{k=1}^{K_i}$ with the action segment start instant $s_k$, the end instant $e_k$ and the corresponding action category $c_k$. TAD aims at detecting all segments $Y_i$ based on the input feature $X_i$.

### 3.1. Method Overview

Our goal is to build a simple and efficient one-stage temporal action detector. As shown in Fig. 3, the overall architecture of TriDet consists of three main parts: a video feature backbone, a SGP feature pyramid, and a boundary-oriented Trident-head. First, the video features are extracted using a pretrained action classification network (e.g. I3D [7] or SlowFast [14]). Following that, a SGP feature pyramid is built to tackle actions with various temporal lengths, similar to some recent TAD works [9, 20, 47]. Namely, the temporal features are iteratively downsampled and each scale level is processed with a proposed Scalable-Granularity Perception (SGP) layer (Section 3.2) to enhance the interaction between features with different temporal scopes. Lastly, action instances are detected by a designed boundary-oriented Trident-head (Section 3.3). We elaborate on the proposed modules in the following.

### 3.2. Feature Pyramid with SGP Layer

The feature pyramid is obtained by first downsampling the output features of the video backbone network several times via max-pooling (with a stride of 2). The features at each pyramid level are then processed using transformer-like layers (e.g. ActionFormer [47]).

Current Transformer-based methods for TAD tasks primarily rely on the macro-architecture of the Transformer (See supplementary material for details), rather than the self-attention mechanisms. Specifically, SA mainly encounters two issues: the rank loss problem across the temporal dimension and its high computational overhead.

**Limitation 1: the rank loss problem.** The rank loss problem arises because the probability matrix in self-attention (i.e. $\text{softmax}(QK^T)$) is non-negative and the sum of each row is 1, indicating the outputs of SA are convex combination for the value feature $V$. Considering that pure Layer Normalization [2] projects feature onto the unit hyper-sphere in high-dimensional space, we analyze the degree of their distinguishability by studying the maximum angle between features within the instant features. We demonstrate that the maximum angle between features after the convex combination is less than or equal to that of the input features, resulting in increasing similarity between features (as outlined in the supplementary material), which can be detrimental to TAD.

**Limitation 2: high computational complexity.** In addition, the dense pair-wise calculation (between instant features) in self-attention brings a high computational overhead and therefore decreases the inference speed.

**The SGP layer.** Based on the above discovery, we propose a Scalable-Granularity Perception (SGP) layer to effectively capture the action information and suppress rank loss. The major difference between the Transformer layer and SGP layer is the replacement of the self-attention module with the fully-convolutional module SGP. The successive Layer Normalization [2] (LN) is changed to Group Normalization [41] (GN).

As shown in Fig. 4, SGP contains two main branches: an instant-level branch and a window-level branch. In the
The resultant SGP-based feature pyramid can achieve better performance than the transformer-based feature pyramid while being much more efficient.

3.3. Trident-head with Relative Boundary Modeling

Intrinsic property of action boundaries. Regarding the detection head, some existing methods directly regress the temporal length [47] of the action at each instant of the feature and refine with the boundary feature [20, 33], or [21, 22, 46] simply predict an actionness score (indicating the probability of being an action). These simple strategies suffer from a problem in practice: imprecise boundary predictions, due to the intrinsic property of actions in videos. Namely, the boundaries of actions are usually not obvious, unlike the boundaries of objects in object detection. Intuitively, a more statistical boundary localization method can reduce uncertainty and facilitate more precise boundaries.

Trident-head. In this work, we propose a boundary-oriented Trident-head to precisely locate the action boundaries based on the relative boundary modeling, i.e. considering the relation of features in a certain period and obtaining the relative probability of being a boundary for each instant in that period. The Trident-head consists of three components: a start head, an end head, and a center-offset head, which are designed to locate the start boundary, end boundary, and the temporal center of the action, respectively. The Trident-head can be trained end-to-end with the detector.

Concretely, as shown in Fig. 5, given a sequence of features \( F \in \mathcal{R}^{T \times D} \) output from the feature pyramid, we first obtain three feature sequences from the three branches (namely, \( F_s \in \mathcal{R}^T, F_e \in \mathcal{R}^T \) and \( F_c \in \mathcal{R}^{T \times 2B} \), where \( B \) is the number of bins for boundary prediction, \( F_s \) and \( F_c \) characterize the response value for each instant as

\[
\begin{align*}
\text{Start Prediction} & \quad \text{Center Offset} & \quad \text{Video Feature} & \quad \text{Reference Times} \\
F_s & \quad F_c & \quad F & \quad F_c^0
\end{align*}
\]

Figure 5. The boundary localization mechanism of Trident-head. We predict the boundary response and the center offset for each instant. At the instant \( t \), the predicted boundary response in neighboring bin set is summed element-wise with the center offset corresponding to the instant \( t \), which is further estimated as the relative boundary distribution. Finally, the offset is computed based on the expected value of the bin.
We apply the feature pyramid levels to reduce the number of parameters. They only consider global information instead of boundary predictions. We divide them into two broad categories: the prediction based on sampling instants in segments \([21,27,35]\) and the prediction based on a single instant. The first category predicts the boundary according to the global feature of the predicted instance segments. They only consider global information instead of detailed information at each instant. The second category directly predicts the distance between an instant and its corresponding boundary based on the instant-level feature \([20,33,47,49]\). Some of them refine the segment with boundary features \([20,33,49]\). However, they do not take the relation (i.e. relative probability of being a boundary) of adjacent instants into account. The proposed Trident-head differs from these two categories and shows superior performance in precise boundary localization.

### 3.4. Training and Inference

Each layer \(l\) of the feature pyramid outputs a temporal feature \(F^l \in \mathcal{R}^{(2^{l-1}T) \times D}\), which is then fed to the classification head and the Trident-head for action instance detection. The output of each instant \(t\) in feature pyramid layer \(l\) is denoted as \(\hat{\theta}^l_t = (\hat{s}^l_t, \hat{d}^l_{st}, \hat{d}^l_{et})\).

The overall loss function is then defined as follows:

\[
\mathcal{L} = \frac{1}{N_{pos}} \sum_{i,t} \mathbb{1}_{\{c^i_t > 0\}}(\sigma_{1:U} \mathcal{L}_{cls} + \mathcal{L}_{reg}) + \frac{1}{N_{neg}} \sum_{i,t} \mathbb{1}_{\{c^i_t = 0\}} \mathcal{L}_{cls},
\]

where \(\sigma_{1:U}\) is the temporal IoU between the predicted segment and the ground truth action instance, and \(\mathcal{L}_{cls}, \mathcal{L}_{reg}\) is focal loss \([23]\) and IoU loss \([34]\). \(N_{pos}\) and \(N_{neg}\) denote the number of positive and negative samples. The term \(\sigma_{1:U}\) is used to reweight the classification loss at each instant, such that instants with better regression (i.e., of higher quality) contribute more to the training. Following previous methods \([39,47,48]\), center sampling is adopted to determine the positive samples. Namely, the instants around the center of an action instance are labeled as positive and all the others are considered as negative.

**Inference.** At inference time, the instants with classification scores higher than threshold \(\lambda\) and their corresponding instances are kept. Lastly, Soft-NMS \([4]\) is applied for the deduplication of predicted instances.

### 4. Experiments

**Datasets.** We conduct experiments on four challenging datasets: THUMOS14 \([17]\), ActivityNet-1.3 \([5]\), HACS-Segment \([50]\) and EPIC-KITCHEN 100 \([11]\). THUMOS14 consists of 20 sport action classes and it contains 200 and 213 untrimmed videos with 3,007 and 3,358 action instances on the training set and test set, respectively. ActivityNet and HACS are two large-scale datasets and they share 200 classes of action. They have 10,024 and 37,613 videos for training, as well as 4,926 and 5,981 videos for test. The EPIC-KITCHEN 100 is a large-scale dataset in first-person vision, which have two sub-tasks: noun localization (e.g. door) and verb localization (e.g. open the door).
Table 1. Comparison with the state-of-the-art methods on THUMOS14 dataset. *: TSN backbone. †: Swin Transformer backbone. Others: I3D backbone.

<table>
<thead>
<tr>
<th>Method</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMN [21]*</td>
<td>56.0</td>
<td>47.4</td>
<td>38.8</td>
<td>29.7</td>
<td>20.5</td>
<td>38.5</td>
</tr>
<tr>
<td>G-TAD [43]*</td>
<td>54.5</td>
<td>47.6</td>
<td>40.3</td>
<td>30.8</td>
<td>23.4</td>
<td>39.3</td>
</tr>
<tr>
<td>A2Net [44]</td>
<td>58.6</td>
<td>54.1</td>
<td>45.5</td>
<td>32.5</td>
<td>17.2</td>
<td>41.6</td>
</tr>
<tr>
<td>TCANet [33]*</td>
<td>60.6</td>
<td>53.2</td>
<td>44.6</td>
<td>36.8</td>
<td>26.7</td>
<td>44.3</td>
</tr>
<tr>
<td>RTD-Net [38]</td>
<td>68.3</td>
<td>62.3</td>
<td>51.9</td>
<td>38.8</td>
<td>23.7</td>
<td>49.0</td>
</tr>
<tr>
<td>VSGN [49]*</td>
<td>66.7</td>
<td>60.4</td>
<td>52.4</td>
<td>41.0</td>
<td>30.4</td>
<td>50.2</td>
</tr>
<tr>
<td>ContextLoc [53]</td>
<td>68.3</td>
<td>63.8</td>
<td>54.3</td>
<td>41.8</td>
<td>26.2</td>
<td>50.9</td>
</tr>
<tr>
<td>AFSD [20]</td>
<td>67.3</td>
<td>62.4</td>
<td>55.5</td>
<td>43.7</td>
<td>31.1</td>
<td>52.0</td>
</tr>
<tr>
<td>ReAct [35]*</td>
<td>69.2</td>
<td>65.0</td>
<td>57.1</td>
<td>47.8</td>
<td>35.6</td>
<td>55.0</td>
</tr>
<tr>
<td>TadTR [27]</td>
<td>74.8</td>
<td>69.1</td>
<td>60.1</td>
<td>46.6</td>
<td>32.8</td>
<td>56.7</td>
</tr>
<tr>
<td>TALLFormer [9]†</td>
<td>76.0</td>
<td>-</td>
<td>63.2</td>
<td>-</td>
<td>34.5</td>
<td>59.2</td>
</tr>
<tr>
<td>ActionFormer [47]</td>
<td>82.1</td>
<td>77.8</td>
<td>71.0</td>
<td>59.4</td>
<td>43.9</td>
<td>66.8</td>
</tr>
<tr>
<td>TriDet</td>
<td>83.6</td>
<td>80.1</td>
<td>72.9</td>
<td>62.4</td>
<td>47.4</td>
<td>69.3</td>
</tr>
</tbody>
</table>

Table 2. Comparison with the state-of-the-art methods on HACS dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>0.5</th>
<th>0.75</th>
<th>0.95</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN [52]</td>
<td>I3D</td>
<td>28.8</td>
<td>18.8</td>
<td>5.3</td>
<td>19.0</td>
</tr>
<tr>
<td>LoFi [42]</td>
<td>TSM</td>
<td>37.8</td>
<td>24.4</td>
<td>7.3</td>
<td>24.6</td>
</tr>
<tr>
<td>G-TAD [43]</td>
<td>I3D</td>
<td>41.1</td>
<td>27.6</td>
<td>8.3</td>
<td>27.5</td>
</tr>
<tr>
<td>TadTR [27]</td>
<td>I3D</td>
<td>47.1</td>
<td>32.1</td>
<td>10.9</td>
<td>32.1</td>
</tr>
<tr>
<td>BMN [21]</td>
<td>SlowFast</td>
<td>52.5</td>
<td>36.4</td>
<td>10.4</td>
<td>35.8</td>
</tr>
<tr>
<td>TALLFormer [9]</td>
<td>Swin</td>
<td>55.0</td>
<td>36.1</td>
<td>11.8</td>
<td>36.5</td>
</tr>
<tr>
<td>TCANet [33]</td>
<td>SlowFast</td>
<td>54.1</td>
<td>37.2</td>
<td>11.3</td>
<td>36.8</td>
</tr>
<tr>
<td>TriDet</td>
<td>I3D</td>
<td>54.5</td>
<td>36.8</td>
<td>11.5</td>
<td>36.8</td>
</tr>
<tr>
<td>TriDet</td>
<td>SlowFast</td>
<td>56.7</td>
<td>39.3</td>
<td>11.7</td>
<td>38.6</td>
</tr>
</tbody>
</table>

It contains 495 and 138 videos with 67,217 and 9,668 action instances for training and test, respectively. The number of action classes for noun and verb are 300 and 97.

Evaluation. For all these datasets, only the annotations of the training and validation sets are accessible. Following the previous practice [9, 21, 46, 47], we evaluate on the validation set. We report the mean average precision (mAP) at different intersection over union (IoU) thresholds. For THUMOS14 and EPIC-KITCHEN, we report the IoU thresholds at [0.3:0.7:0.1] and [0.1:0.5:0.1] respectively. For ActivityNet and HACS, we report the result at IoU thresholds [0.5, 0.75, 0.95] and the average mAP is computed at [0.5:0.95:0.05].

4.1. Implementation Details

TriDet is trained end-to-end with AdamW [31] optimizer. The initial learning rate is set to $10^{-4}$ for THUMOS14 and EPIC-KITCHEN, and $10^{-3}$ for ActivityNet and HACS. We detach the gradient before the start boundary head and end boundary head and initialize the CNN weights of these two heads with a Gaussian distribution $\mathcal{N}(0, 0.1)$ to stabilize the training process. The learning rate is updated with Cosine Annealing schedule [30]. We train 40, 23, 19, 15 and 13 epochs for THUMOS14, EPIC-KITCHEN verb, EPIC-KITCHEN noun, ActivityNet and HACS (containing warmup 20, 5, 5, 10, 10 epochs).

For ActivityNet and HACS, the number of bins $B$ of the Trident-head is set to 12, 14 and the convolution window $w$ is set to 15, 11 and the scale factor $k$ is set to 1.3 and 1.0, respectively. For THUMOS14 and EPIC-KITCHEN, the number of bins $B$ of the Trident-head is set to 16 and the convolution window $w$ is set to 1 and the scale factor $k$ is set to 1.5. We round the scaled windows size and take it up to the nearest odd number for convenience. We conduct our experiments on a single NVIDIA A100 GPU.

4.2. Main Results

THUMOS14. We adopt the commonly used I3D [7] as our backbone feature and Tab. 1 presents the results. Our method achieves an average mAP of 69.3%, outperforming all previous methods including one-stage and two-stage methods. Notably, our method also achieves better performance than recent Transformer-based methods [9, 27, 33, 35, 47], which demonstrates that the simple design can also have impressive results.

HACS. For the HACS-segment dataset, we conduct experiments based on two commonly used features: the official I3D [7] feature and the SlowFast [14] feature. As shown in Tab. 2, our method achieves an average mAP of 36.8% with the official features. It is the state-of-the-art and outperforms the previous best model TadTR by about 4.7% in average mAP. We also show that changing the backbone to SlowFast can further boost performance, resulting in a 1.8% increase in average mAP, which indicates that our method can benefit from a much more advanced backbone network.

EPIC-KITCHEN. On this dataset, following all previous methods, SlowFast is adopted as the backbone feature.
Table 4. Comparison with the state-of-the-art methods on ActivityNet-1.3 dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>0.5</th>
<th>0.75</th>
<th>0.95</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGCN [46]</td>
<td>3D</td>
<td>48.3</td>
<td>33.2</td>
<td>3.3</td>
<td>31.1</td>
</tr>
<tr>
<td>ReAct [35]</td>
<td>TSN</td>
<td>49.6</td>
<td>33.0</td>
<td>8.6</td>
<td>32.6</td>
</tr>
<tr>
<td>BMN [21]</td>
<td>TSN</td>
<td>50.1</td>
<td>34.8</td>
<td>8.3</td>
<td>33.9</td>
</tr>
<tr>
<td>G-TAD [43]</td>
<td>TSN</td>
<td>50.4</td>
<td>34.6</td>
<td>9.0</td>
<td>34.1</td>
</tr>
<tr>
<td>AFSD [20]</td>
<td>3D</td>
<td>52.4</td>
<td>35.2</td>
<td>6.5</td>
<td>34.3</td>
</tr>
<tr>
<td>TadTR [27]</td>
<td>TSN</td>
<td>51.3</td>
<td>35.0</td>
<td>9.5</td>
<td>34.6</td>
</tr>
<tr>
<td>TadTR [27]</td>
<td>R(2+1)D</td>
<td>53.6</td>
<td>37.5</td>
<td>10.5</td>
<td>36.8</td>
</tr>
<tr>
<td>VSGN [49]</td>
<td>3D</td>
<td>52.3</td>
<td>35.2</td>
<td>8.3</td>
<td>34.7</td>
</tr>
<tr>
<td>PBRNet [24]</td>
<td>3D</td>
<td>54.0</td>
<td>35.0</td>
<td>9.0</td>
<td>35.0</td>
</tr>
<tr>
<td>TCANet+BMN [33]</td>
<td>TSN</td>
<td>52.3</td>
<td>36.7</td>
<td>6.9</td>
<td>35.5</td>
</tr>
<tr>
<td>TCANet+BMN [33]</td>
<td>SlowFast</td>
<td>54.3</td>
<td>39.1</td>
<td>8.4</td>
<td>37.6</td>
</tr>
<tr>
<td>TALLFormer [9]</td>
<td>Swin</td>
<td>54.1</td>
<td>36.2</td>
<td>7.9</td>
<td>35.6</td>
</tr>
<tr>
<td>ActionFormer [47]</td>
<td>R(2+1)D</td>
<td>54.7</td>
<td>37.8</td>
<td>8.4</td>
<td>36.6</td>
</tr>
<tr>
<td>TriDet</td>
<td>R(2+1)D</td>
<td>54.7</td>
<td>38.0</td>
<td>8.4</td>
<td>36.8</td>
</tr>
</tbody>
</table>

The method of our main comparison is ActionFormer [47], which has demonstrated promising performance in EPIC-KITCHEN 100 dataset. We present the results in Tab. 3. Our method shows a significant improvement in both sub-tasks: *verb* and *noun*, and achieves 25.4% and 23.8% average mAP, respectively. Note that our method outperforms ActionFormer with the same features by a large margin (1.9% and 1.9% average mAP in *verb* and *noun*, respectively). Moreover, our method achieves state-of-the-art performance on this challenging dataset.

**ActivityNet.** For the ActivityNet v1.3 dataset, we adopt the TSP R(2+1)D [1] as our backbone feature. Following previous methods [9, 20, 27, 33, 47], the video classification score predicted from the UntrimmedNet is adopted to multiply the final detection score. Tab. 4 presents the results. Our method still shows a promising result: TriDet outperforms the second best model [47] with the same feature, only worse than TCANet [33] which is a two-stage method and using the SlowFast as the backbone feature which is not available now.

### 4.3. Ablation Study

In this section, we mainly conduct the ablation studies on the THUMOS14 dataset.

**Main components analysis.** We demonstrate the effectiveness of our proposed components in TriDet: SGP layer and Trident-head. To verify the effectiveness of our SGP layer, we use a baseline feature pyramid used by [20, 47] to replace our SGP layer. The baseline consists of two 1D-convolutional layers and shortcut. The window size of convolutional layers is set to 3 and the number of channels of the intermediate feature is set to the same dimension as the intermediate dimension in the FFN in our SGP layer. All other hyperparameters (e.g. number of the pyramid layers, etc.) are set to the same as our framework.

As depicted in Tab. 5, compared with the baseline model we implement (Row 1), the SGP layer brings a 6.2% absolute improvement in the average mAP. Secondly, we compare the SGP with the previous state-of-the-art method, ActionFormer, which adopts a self-attention mechanism in a sliding window behavior [3] with window size 7 (Row 2). We can see our SGP layer still has 1.5% improvement in average mAP, demonstrating that the convolutional network can also have excellent performance in TAD task. Besides, we compare our Trident-head with the normal instant-level regression head, which regresses the boundary distance for each instant. We can see that the Trident-head improves the average mAP by 1.0%, and the mAP improvement is more obvious in the case of high IoU threshold (e.g. 1.6% average mAP improvement in IoU 0.7).

**Computational complexity.** We compare the computational complexity and latency of TriDet with the recent ActionFormer [47], which brings a large improvement to TAD by introducing the Transformer-based feature pyramid.

As shown in Tab. 6, we divide the detector into two parts: the main architecture and the detection heads (e.g. classification head and regression head). We report the GMACs for each part and the inference latency (average over five times) on THUMOS14 dataset using an input with the shape 2304 × 2048, following the [47]. We also report our results using the Trident-head and the normal regression head, respectively. First, from the first row, we see that GMACs of our main architecture with SGP layer is only 47.1% of the total GMACs.
the ActionFormer (14.5 versus 30.8), and the overall latency is only 65.2% (146ms versus 224ms), but TriDet still outperforms ActionFormer by 1.5% average mAP, which shows that our main architecture is much better than the local Transformer-based method. Besides, we further evaluate our method with Trident-head. The experimental result shows that our framework can be improved by the Trident-head which further brings 1.0% average mAP improvement and the GMACs is still 1.6G smaller than ActionFormer, and the latency is still only 74.6% of it, proving the high efficiency of our method.

**Ablation on the window size in SGP layer.** In this section, we study the effectiveness of the two hyper-parameters related to the window size in the SGP layer. Firstly, we fix \( k = 1 \) and vary \( w \). Secondly, we fix the value of \( w = 1 \) and change \( k \). Finally, we present the results in Fig. 6 on THUMOS14 datasets. We find that different choices of \( w \) and \( k \) produce stable results on both datasets. The optimal values are \( w = 1, k = 5 \) for THUMOS14.

**The effectiveness of feature pyramid level.** To study the effectiveness of the feature pyramid and its relation with the number of Trident-head bin set, we start the ablation from the feature pyramid with 16 bins and 6 levels. We conduct two sets of experiments: a fixed number of bins or a scaled number of bins for each level in the feature pyramid. As shown in Tab. 7, we can see that the detection performance rises as the number of layers increases. With fewer levels (i.e. level less than 3), more bins bring better performance. That is because the fewer the number of levels, the more bins are needed to predict the action with a long duration (i.e. higher resolution at the highest level). We achieve the best result with a level number of 6.

**Ablation on the number of bins.** In this section, we present the ablation results for the choice of the number of bins on the THUMOS14 and HACS datasets in Tab. 8. We observe the optimal value is obtained at 16 and 14 on the THUMOS14 and the HACS, respectively. We also find that a small bin value leads to significant performance degradation on HACS but not on THUMOS14. That is because the THUMOS14 dataset aims at detecting a large number of action segments from a long video and a small bin value can meet the requirements, but on HACS, there are more actions with long duration, thus a larger number of bins is needed.

**5. Conclusion**

In this paper, we aim at improving the temporal action detection task with a simple one-stage convolutional-based framework TriDet with relative boundary modeling. Experiments conducted on THUMOS14, HACS, EPIC-KITCHEN and ActivityNet demonstrate a high generalization capability of our method, which achieves state-of-the-art performance on the first three datasets and comparable results on ActivityNet. Extensive ablation studies are conducted to verify the effectiveness of each proposed component.

**Acknowledgement.** This work is supported by the National Natural Science Foundation of China under Grant 62132002.
References


